Possibilistic WorkFlow nets to deal with non-conformance in Process Execution

Leiliane Pereira de Rezende*, Stéphane Julia* and Janette Cardoso†
*Faculdade de Computação - Universidade Federal de Uberlândia, UFU
2160, av. João Naves de Ávila, 38400-902 Uberlândia/MG, Brazil
Email: leily.rezende@gmail.com, stephane@facom.ufu.br
† Institut Supérieur de l'Aéronautique et de l'Espace,ISAE
10, av. Édouard Belin, 31055 Toulouse, France
Email: cardoso@isae.fr

***Abstract*—In this paper, an approach based on WorkFlow nets and on possibilistic Petri nets is proposed to deal with non-conformance in Business Processes. Routing patterns existing in Business Process are modeled by WorkFlow nets. To express in a more realistic way the uncertainty attached to human activities, possibilistic Petri nets with uncertainty on the marking and on the transition firing are considered. Combining both formalisms, a kind of possibilistic WorkFlow net is obtained. An example of deviation at a process monitoring level due to human behavior in a "Handle Complaint Process" is presented.**

***Index Terms*—WorkFlow net, possibilistic Petri net, process non-conformance, process monitoring.**

I. Introduction

The purpose of Workflow Management Systems [1] is to execute Business Processes. Business Processes represent the sequences of activities that have to be executed within an organization to treat specific cases and to reach well defined goals. Over the last few years, Business Process Management has become important in order to raise service quality and performance of firms [2].

Many papers have already considered Petri net theory as an efficient tool for the modeling and analysis of Workflow Management Systems. In [1], WorkFlow nets, which are acyclic Petri net models used to represent Business Processes, are defined.

Soundness property is an important criterion which needs to be satisfied when treating Workflow Processes. In fact, good properties of well defined formal models such as WorkFlow nets can easily be proved when Business Processes are following a rigid structure that does not allow deviations from the process description during the real time execution. However, recently, it was shown that Business Processes do not easily map to a rigid modeling structure. As a matter of fact, Business Process Implementation depends on human work. Tasks performed by humans are generally complex and follow rules that cannot always be transformed into computerized processes.

Attempts to consider a certain level of flexibility in process definition has already been proposed by several authors.

In [2], the Yawl language which supports flexibility in the process definition is proposed. The principle is based on Worklet Services which allow for the construct of subprocess structures in such a way that they can be added dynamically to the whole Workflow Process during the real time execution. In such an approach, the possible process deviations are designed in an explicit way (additional routing structures in the model) and no guarantee of soundness in the process exists.

In [3], a deviation-tolerant approach for software processes is presented. Here, two models coexist during the monitoring of the process. One corresponds to the normal behavior and can be seen as a previsional process model. The other process model is dynamically built through the observation of actions of human actors. The two models are then permanently compared to detect possible deviations. The difficulty in such an approach is to deal simultaneously with two models that can overload the monitoring activity and reduce the performance of the system.

In [4], a kind of declarative implicit model, essentially based on rules, is used to detect inconsistencies (non-conformant states in the process) and to accept possible deviations (non-conformant transitions between activities). The limitation of the methodology is related to the process model that can be seen as a simple set of constraints without a real process structure that can be analysed from the point of view of basic properties, such as soundness.

In [5] and [6], a process model based on temporal logic and finite state machines to capture and tolerate deviations in processes during execution is defined. For the authors, a process is correct if all the constraints given by the set of state machines are verified. In particular, two kinds of transition are created: normal ones and exported ones which depend on user requests to indicate abnormal behavior. In particular, when the process execution is corrupted, the state of the process is fixed manually. The problem with this kind of approach is again that the model of the process is given through a declarative form instead of a single graph representing a whole process that could be analysed from the point of view of soundness property, as it is the case with WorkFlow nets. Another problem is the necessity for explicitly model alternative scenarios corresponding to abnormal behaviors. The consequence is generally the increase of the complexity of the

set of constraints and on the underlying process model.

A very promising alternative to deal with flexibility in Business Processes seems to be approaches based on uncertain reasoning as that presented in [7]. The model of the process is then given through fuzzy sets and possibilistic distributions. The advantage of fuzzy and possibility reasoning is the ability to naturally represent uncertain and imprecise information that exists when activities are executed by human employees.

One of the first studies which combines fuzzy and possibilistic representation of information with the precise structure of a Petri net when considering discrete event systems is the one described in [8] and [9]. The main feature of possibilistic/fuzzy Petri nets is to allow one to reason about the aspects of uncertainty and change in dynamic discrete event systems. Most of the examples presented by the authors of possibilistic Petri net were applied to flexible manufacturing systems.

In this paper, an approach based on WorkFlow nets and possibilistic Petri nets is proposed to deal with flexibility in Business Processes. In particular, a kind of possibilistic WorkFlow net will be defined to treat deviation as well as inconsistencies in Business Processes.

In section II, the definition of WorkFlow nets and soundness correctness criterion are provided. In section III, an overview of possibilistic Petri net is given. In section IV, a new model of Business Processes is defined: the possibilistic WorkFlow net. An example based on a "Handle Complaint Process" illustrates the approach. Finally, the last section concludes this work with a short summary, an assessment about the approach presented and an outlook on the future work.

II. WorkFlow Nets

A Petri net that models a Workflow Process is called a WorkFlow net [1], [10]. A WorkFlow net satisfies the following properties [10]:

- It has only one source place, named *Start* and only one sink place, named *End*. These are special places such that the place *Start* has only outgoing arcs and the place *End* has only incoming arcs.
- A token in *Start* represents a case that needs to be handled and a token in *End* represents a case that has been handled.
- Every task t (transition) and condition p (place) should be on a path from place *Start* to place *End*.

Soundness is a correctness criterion defined for WorkFlow nets. A WorkFlow net is sound if, and only if, the following three requirements are satisfied [1]:

- For each token put in the place *Start*, one and only one token appears in the place *End*.
- When the token appears in the place *End*, all the other places are empty for this case.
- For each transition (task), it is possible to move from the initial state to a state in which that transition is enabled, i.e. there are not any dead transitions.

A method for the qualitative analysis of WorkFlow nets (soundness verification) based on the proof trees of linear logic is presented in [11].

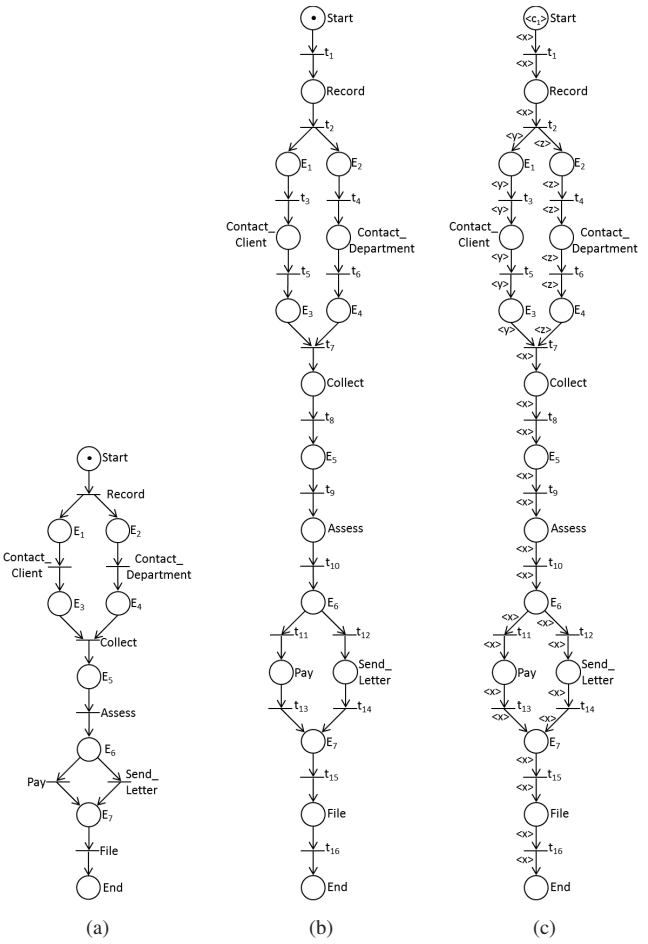


Fig. 1. Handle Complaint Process: (a)Tasks are associated directly to simple transitions. (b)Tasks are associated directly to places. (c)Possibilistic WorkFlow net

A. Process

A process defines which tasks need to be executed and in which order [10]. Modeling a Workflow Process in term of a WorkFlow net is rather straightforward: transitions are active components and models the tasks, places are passive components and models conditions (pre and post) and tokens model cases [1], [10]. To illustrate the mapping of a process into WorkFlow nets, the process for handling complaints that is shown in [1] can be considered as follows: an incoming complaint is first recorded. Then the client who has complained and the department affected by the complaint are contacted. The client is approached for more information. The department is informed of the complaint and may be asked for its initial reaction. These two tasks may be performed in parallel, i.e. simultaneously or in any order. After this, the data is gathered and a decision is made. Depending upon the decision, either a compensation payment is made or a letter is sent. Finally, the complaint is filed. Fig. 1(a) shows a WorkFlow net that correctly models this process.

B. Routing constructs

Tasks can be optional, i.e. there are tasks that just need to be executed for some cases, and the order in which tasks will be executed can vary from case to case [1]. Four basic constructions for routing are presented in [1] and [10]:

- *Sequential*: tasks are executed one after another sequentially, clearly demonstrating dependence among these tasks: one needs to finish for the other to start;
- *Parallel*: if more than one task can be executed simultaneously or in any order. In this case, both tasks can be executed without the result of one interfering in the result of the other;
- *Conditional* (or selective routing): when there is a choice between two or more tasks;
- *Iterative*: when it is necessary to execute the same task multiple times.

Some variations of these four basic constructions can be found in [1] and [10].

Considering the "Handle Complaint Process" shown in Fig. 1(a), tasks "Contact Client" and "Contact Department" are an example of parallel routing. Tasks "Collect" and "Assess" are an example of sequential routing. And tasks "Pay" and "Send Letter" are an example of conditional routing.

C. Process Monitoring

In [2], WorkFlow nets were revisited in terms of their suitability for monitoring Business Processes. The authors showed that some patterns were not easily captured, in particular patterns dealing with cancellation and multiple concurrently executing instances of the same task.

The principal reason of limitation existing in WorkFlow nets for monitoring Business Processes is the fact that tasks are associated directly with simple transitions. As a consequence, once initiated, a task cannot be interrupted because it corresponds to the firing of a transition. If during the execution of a task, an event occurs in the system whose purpose is to interrupt the whole process, in traditional WorkFlow nets the current tasks of the process have to be completed first to be able to accept the cancellation. Of course, a proper model of the process should be able to accept interruption events in an asynchronous way in order to monitor the process in a efficient way.

The solution proposed in this work to deal with interruption during tasks execution is to transform the transitions of WorkFlow net into a structure based on the following pattern: a block corresponding to a task of a transition t_i is composed of a place P_{t_i} which represents the task t_i, an input transition $t_{in}{}_i$ which represents the beginning of the task execution, and an output transition $t_{out}{}_i$ which represents the end of the task execution. The WorkFlow net of Fig. 2(a) will then be transformed into the Workflow Process given by the acyclic Petri net model of Fig. 2(b). As the new block (corresponding to the task in execution) can be substituted by a simple transition preserving the good properties of the initial model [12], the new process model will continue sound in most cases and will be adapted for monitoring activities, in particular if some events of cancellation need to be specified as shown in Fig. 2(c).

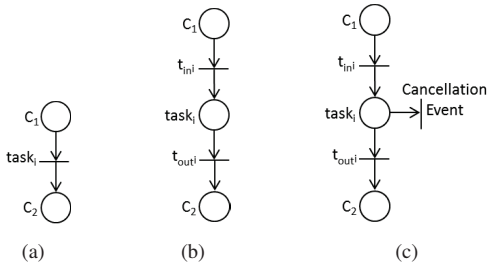


Fig. 2. (a) Traditional WorkFlow net; (b) WorkFlow net with explicit task execution; and (c) WorkFlow net with cancellation event

Finally, the Petri net model of Fig 1(b), corresponding to the WorkFlow net of Fig. 1(a), will then be produced.

III. Possibilistic Petri Nets

In the approach presented in [8], a possibilistic Petri net is a model where a place denotes a possible partial state, a transition denotes a possible state change, and a firing sequence denotes a possible behavior. The main advantage is to be able to update a system state at a supervisory level with ill-known information without having inconsistent states.

A possibilistic Petri net model associates a possibility distribution $\Pi_o(p)$ to the location of an object o, p being a place in the net. As a matter of fact, possibilistic Petri nets are derived from Object Petri nets [13] where tokens are represented by objects with attributes and conditions are associated to transitions. Possibilistic distribution allows one to model:

- *A precise marking*: each token is located in only one place (well-known state).
- *An imprecise marking*: each token location has a possibility distribution over a set of places. It cannot be asserted that a token is in a given place, but only that it is in a place among a given set of places.

$\Pi_o(p) = 1$ represents the fact that p is a possible location of o, and $\Pi_o(p) = 0$ expresses the certainty that o is not present in place p. Formally, a marking in a possibilistic Petri net is then a mapping:

$$M : O \times P \longrightarrow \{0, 1\} \tag{1}$$

where O is a set of objects and P a set of places. If $M(o, p) = 1$, there exists a possibility of having the object o in place p. On the contrary ($M(o, p) = 0$), there is no possibility of having o in p. A marking M of the net allows one to represent:

- *A precise marking*: $M(o, p) = 1$ and $\forall p_i \neq p, M(o, p_i) = 0$.
- *An imprecise marking*: for example, if there exists a possibility at a certain time to have the same object o in two different places, p_1 and p_2, them $M(o, p_1) = M(o, p_2) = 1$.

A possibilistic marking will correspond in practice to knowledge concerning a situation at a given time. In a possibilistic Petri net, the firing (certain or uncertain) of a transition t is decomposed into two steps:

- *Beginning of a firing*: tokens are put into output places of t but are not removed from its input places.
- *End of a firing*: that can be a firing cancellation (tokens are removed from the output places of t) or a firing achievement (tokens are removed from the input places of t).

A certain firing consists of a beginning of a firing and an immediate firing achievement. A pseudo-firing that will increase the uncertainty of the marking can be considered only as the beginning of a firing (there is no information to be sure whether the normal event associated with the transition has actually occurred or not). To a certain extent, pseudo-firing is a way of realizing forward deduction.

The interpretation of a possibilistic Petri net is defined by attaching to each transition an authorization function $\eta_{x_1,...,x_n}$ that represents an extra-firing condition:

$$\eta_{x_1,...,x_n} : T \longrightarrow \{False, Uncertain, True\} \qquad (2)$$

where $x_1, ..., x_n$ are the variables associated to the incoming arcs of transition t (when considering the underlying Object Petri net).

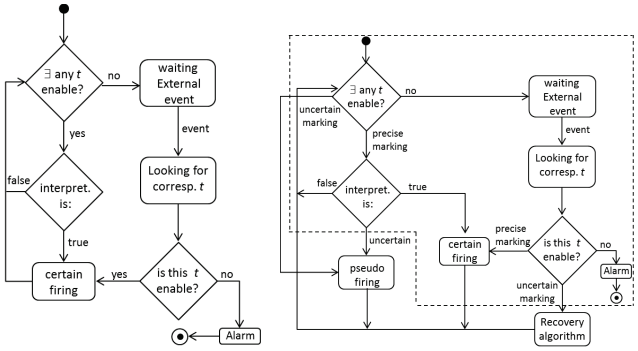
If $o_1, ..., o_n$ is a possible substitution to $x_1, ..., x_n$ for firing t, then several situations can be considered:

- t is not enabled by the marking but the associated interpretation is true; a forbidden situation occurs and an alarm is activated;
- t is enabled by a precise marking and the interpretation is true; then a classical firing (with certainty) of an object Petri net occurs;
- t is enabled by a precise marking and the interpretation is uncertain; then the transition is pseudo-fired and the imprecision is increased;
- t is enabled by an uncertain marking; if the interpretation is uncertain, t is pseudo-fired;
- t is enabled by an uncertain marking and the interpretation is true: a recovery algorithm, presented in [14], is called and a new computation of the possibility distribution of the objects involved in the uncertain marking is realized in order to go back to a certain marking.

Concepts about possibilistic Petri nets will be illustrated through a practical example in the next section.

IV. Possibilistic WorkFlow Nets

If a Petri net is used as a model for Business Processes in a Workflow Management System, transitions will represent the state changes of the process. In particular, each event occurring during the execution of the process (beginning and ending of activities) will be associated with a transition as a boolean variable. Such a variable will be essentially seen as an external value corresponding to a message received from an activity



(a) (b)

Fig. 3. (a) Token player of Petri net. (b) Token player of Possibilistic Petri net

(or send to an activity). Possibly, internal values depending on certain token attributes will enable some transitions too.

Petri net models can be directly executed using a specialized inference mechanism called "*token player* algorithm" that allows for a simplified monitoring of the represented processes, as the one depicted by the activity diagram in Fig. 3(a).

As pointed out in the introduction, the interaction of human behavior in Processes Management can introduce some uncertainty and should be taken into account in the model of the process. A classical token player algorithm, as the one in Fig 3(a), is only based on normal expected events. If an unexpected event occurs, an immediate inconsistency between the model of the process and the real process execution will happen (an external event received by a transition which is not enabled by the marking of the net) or, if some expected event never occurs, the model will reach a deadlock situation (an external event never received by an enabled transition).

A model of the process based on the routing structure of WorkFlow nets and on uncertain marking and firing of possibilistic Petri nets will then produce a kind of possibilistic WorkFlow net that will be able to deal with non-conformance in Business Processes monitoring. The "Handle Complaint Process" represented in Fig. 1(b) will be used to illustrate the approach.

The possibilistic WorkFlow net with objects in Fig. 1(c) represents the new model of the "Handle Complaint Process" where $< c_1 >$ is an object belonging to the class "Complaint", x, y and z are variables of the same class "Complaint" and all places of the model belong to the class "Complaint" too.

After the Complaint $< c_1 >$ is recorded in place "Recorded", the model indicates that copies ($< c_{11} >$ and $< c_{12} >$) of the complaint $< c_1 >$ have to be produced in places E_1 and E_2 to initiate simultaneously the activities "Contact Client" (A_2) and "Contact Department" (A_3). The expected behavior of the process is to have both activities performed in order to continue the treatment of the complaint through the firing of transition t_7. But a deviation of the expected behavior can easily occur if, for example, the client cannot be reached. Normally, the purpose of the activity "Contact Client" is to collect more information from the client

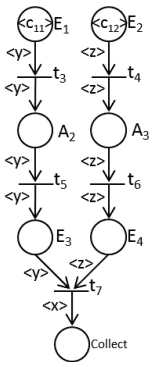


Fig. 4. Fraction of the "Handle Complaint Process"

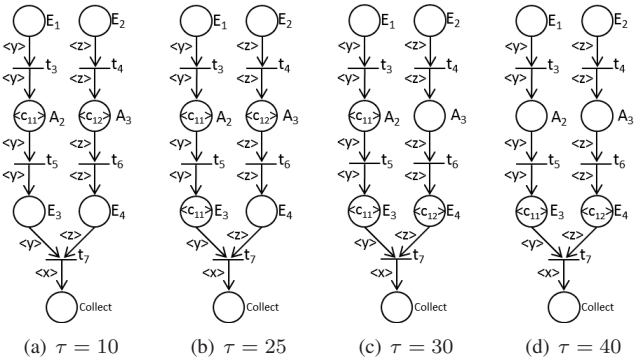


Fig. 5. Simulation results for scenario 1

to help him succeed in resolving its complaint. Logically, if the client cannot be reached, after a while, the process should continue (firing of transition t_7).

If after the firing of transition t_2, copies $< c_{11} >$ and $< c_{12} >$ of the complaint $< c_1 >$ have been produced in places E_1 and E_2, the fraction of the process concerned with possible deviation problems will be the one represented in Fig. 4.

Object instances of class "Complaint" have an attribute $date$. The interpretations of transitions t_5, t_6 and t_7 are given by the following distributions:

$$\eta_y(t_5) = \begin{cases} uncertain & if(\tau \geq y.data) \wedge (\neg endCC(y)) \\ true & if(endCC(y)) \\ false & otherwise \end{cases} \tag{3}$$

$$\eta_z(t_6) = \begin{cases} uncertain & if(\tau \geq z.data) \wedge (\neg endCD(z)) \\ true & if(endCD(z)) \\ false & otherwise \end{cases} \tag{4}$$

$$\eta_{yz}(t_7) = \begin{cases} true & if(\tau \geq max(y.date, z.date)) \\ false & otherwise \end{cases} \tag{5}$$

where τ is the current time, $endCC(y)$ is true when the actor responsible for the activity contact client informs the end of the activity and $endCD(z)$ is true when the actor responsible for the activity contact department informs the end of the activity.

The normal expected behavior of the process corresponds to ending messages received from the actors responsible from the activities (contact department and contact client) before the current time reaches the values indicated by the attributes date associated to the objects c_{11} and c_{12}. If the ending messages are received in time, all the transition firing will be certain and all the markings will be precise.

Abnormal behavior happens when the current time reaches the value of the attribute date of one of the objects c_{11} or c_{12} and no ending message has been received from the actor responsible for the corresponding activity. In this case

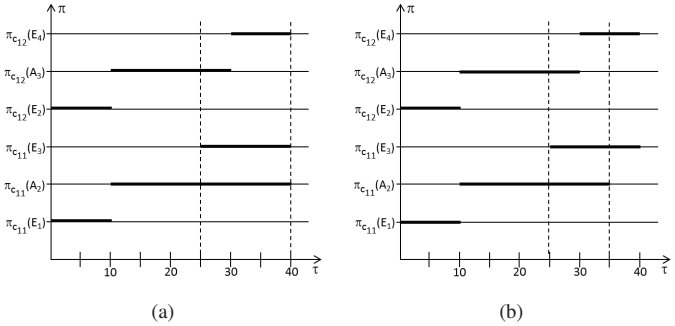


Fig. 6. Possibility distributions of object locations c_{11} and c_{12}: (a) scenario 1; (b) scenario 2

some pseudo-firing will occur and the imprecision about some objects will increase.

Let's assume the transition t_3 and t_4 are fired at date $\tau = 10$ (Fig. 5(a)) and that the actions associated to these transitions are the following:

$$Action(t_3) : y.date = \tau + 15 \tag{6}$$

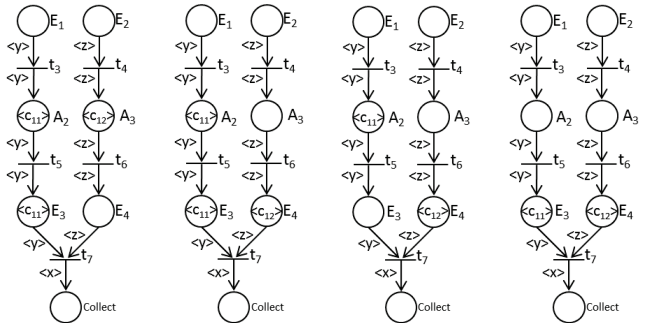
$$Action(t_4) : z.date = \tau + 30 \tag{7}$$

if $endCD(c_{12}) = true$ at $\tau = 30$ and $endCC(c_{11}) = false$ all the time, the following scenario will occur:

- at time $\tau = 25$, $\eta_{c_{11}}(t_5) = uncertain$ and t_5 is pseudo-fired (Fig. 5(b));
- at time $\tau = 30$, $\eta_{c_{12}}(t_6) = true$ and t_6 is fired (Fig. 5(c));
- at time $\tau = 40$, $\eta_{c_{11}c_{12}}(t_7) = true$. t_7 is then enabled by an uncertain marking with a true interpretation. Consequently a recovery algorithm is called to go back to the certain marking of Fig. 5(d).

This scenario corresponds to a situation where the time constraints will not permit an answer from the client. Even so, no inconsistency is detected and the process can proceed. Fig. 6(a) depicts the possibility distributions of instances c_{11} and c_{12} as functions of time (the black lines represent a possibility equal to 1 and the bright lines a possibility equal to 0).

Another scenario could be the following: $endCD(c_{12}) = true$ at $\tau = 30$ and $endCC(c_{11}) = true$ at $\tau = 35$. The following scenario will occur:



(a) $\tau = 25$ (b) $\tau = 30$ (c) $\tau = 35$ (d) $\tau = 35$

Fig. 7. Simulation results for scenario 2

- at time $\tau = 25$, $\eta_{c_{11}}(t_5) = uncertain$ and t_5 is pseudo-fired (Fig. 7(a));
- at time $\tau = 30$, $\eta_{c_{12}}(t_6) = true$ and t_6 is fired (Fig. 7(b));
- at time $\tau = 35$, $\eta_{c_{11}}(t_5) = true$. t_5 is then enabled by an uncertain marking with a true interpretation. Consequently a recovery algorithm is called to go back to the certain marking of Fig. 7(c). Finally, t_5 is fired (Fig. 7(d));
- at time $\tau = 40$, $\eta_{c_{11}c_{12}}(t_7) = true$ and t_7 can be fired.

This scenario corresponds to a situation where the time constraints are tolerant enough to permit an answer from the client before the normal propagation of the process. Fig. 6(b) depicts the possibility distributions of instances c_{11} and c_{12} for this scenario.

The general behavior of a Workflow Management System based on possibilistic WorkFlow net models will be based on the token player given by the activity diagram in Fig. 3(b). It is easy to see that this possibilistic WorkFlow net token player encompasses the expected behavior of the process as well as tolerable deviations.

To take into account the kind of incident presented in this part with an ordinary Petri net based on the token player in Fig. 3(a), several new transitions should be created to consider all possible abnormal scenarios. As a consequence, the corresponding graph would rapidly become completely unreadable.

V. Conclusion

In this article, a new possibilistic WorkFlow net model was presented with the purpose of dealing with non-conformance in Business Processes. Combining the routing structure of WorkFlow nets with the uncertain reasoning of possibilistic Petri nets, it was possible to accept tolerable deviations from the process description without necessarily reaching state inconsistencies. Such a model was applied to a "Handle Complaint Process" when human actors are involved in the process activities.

Comparing this approach with other works dealing with the problem of non-conformance, the main advantage is the fact that a formal process model allowing for the proving of some good properties, like the soundness property for example, was combined with a possibilistic approach very well adapted to the concept of flexibility in processes.

As a future work proposal, it will be interesting to combine the structure of a WorkFlow net with a Time Fuzzy Petri net as the one presented in [8]. The use of possibilistic distributions based on time fuzzy sets should be able to produce a kind of hierarchy between a set of possible alternatives turning some events more plausible than others. Such an approach should be interesting at the monitoring level in order to diminish the imprecision of human intervention.

Acknowledgment

The authors would like to thank FAPEMIG, CAPES and CNPq for financial support.

References

[1] W. v. d. Aalst and K. v. Hee, *Workflow Management: Models, Methods, and Systems*, 1st ed., ser. MIT Press Books. The MIT Press, 2004, vol. 1.

[2] A. Hofstede, W. van der Aalst, M. Adams, and N. Russell, *Modern Business Process Automation: YAWL and its Support Environment*, 1st ed. Springer Publishing Company, Incorporated, 2009.

[3] K. Mohammed, L. Redouane, and C. Bernard, "A deviation-tolerant approach to software process evolution," in *Ninth international workshop on Principles of software evolution: in conjunction with the 6th ESEC/FSE joint meeting*, ser. IWPSE '07. New York, NY, USA: ACM, 2007, pp. 75–78.

[4] S. Thompson and T. Torabi, "An observational approach to practical process non-conformance detection," in *Applications of Digital Information and Web Technologies, 2009. ICADIWT '09. Second International Conference on the*, aug. 2009, pp. 62 –67.

[5] G. Cugola, E. Di Nitto, C. Ghezzi, and M. Mantione, "How to deal with deviations during process model enactment," in *Proceedings of the 17th international conference on Software engineering*, ser. ICSE '95. New York, NY, USA: ACM, 1995, pp. 265–273.

[6] G. Cugola, E. Di Nitto, A. Fuggetta, and C. Ghezzi, "A framework for formalizing inconsistencies and deviations in human-centered systems," *ACM Trans. Softw. Eng. Methodol.*, vol. 5, no. 3, pp. 191–230, jul 1996.

[7] S. Cîmpan and F. Oquendo, "Dealing with software process deviations using fuzzy logic based monitoring," *SIGAPP Appl. Comput. Rev.*, vol. 8, no. 2, pp. 3–13, dec 2000.

[8] J. Cardoso, *Time Fuzzy Petri Nets*, ser. Studies in Fuzziness and Soft Computing. Physica-Verlag, 1999, vol. 22, pp. 115–145.

[9] J. Cardoso, R. Valette, and D. Dubois, "Possibilistic petri nets," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 29, no. 5, pp. 573 –582, oct 1999.

[10] W. M. P. Van Der Aalst, "The application of petri nets to workflow management," *Journal of Circuits Systems and Computers*, vol. 8, no. 1, pp. 21–66, 1998.

[11] L. Soares Passos and S. Julia, "Qualitative analysis of workflow nets using linear logic: Soundness verification," in *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, oct. 2009, pp. 2843 –2847.

[12] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541 – 580, April 1989.

[13] C. Sibertin-Blanc, "Cooperative objects: Principles, use and implementation," in *Concurrent Object-Oriented Programming and Petri Nets*, ser. Lecture Notes in Computer Science, G. Agha, F. De Cindio, and G. Rozenberg, Eds. Springer Berlin / Heidelberg, 2001, vol. 2001, pp. 216–246.

[14] J. Cardoso, R. Valette, and D. Dubois, "Petri nets with uncertain markings," in *Advances in Petri Nets 1990*, ser. Lecture Notes in Computer Science, G. Rozenberg, Ed. Springer Berlin / Heidelberg, 1991, vol. 483, pp. 64–78.